This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (Currently Amended) A method for loading input data in one or more hierarchical format input files into a data store, comprising:

generating a map specification that maps input data in the one or more input files to columns of tuples;

performing parallel processing of the one or more input files to output ~~data~~ tuples, wherein the parallel processing includes parallel parsing and construction of the tuples using the map specification; and

serially loading the ~~data~~ tuples into the data store while enforcing the order of the data in the one or more input files.

2. (Original) The method of claim 1, further comprising:

receiving a physical input file; and

logically dividing the physical input file into multiple sections, wherein each of the multiple sections is an input file.

3. (Original) The method of claim 2, further comprising:

while performing processing of a first section from the multiple sections under control of a first row mapper,

determining that there has been an error in logically dividing the physical input file;

continuing processing of a next section from the multiple sections that is also being processed by a second row mapper; and

notifying the second row mapper to terminate processing of the next section.

4. (Currently Amended) The method of claim 3, wherein the tuples output when processing data from each of the input files [[is]] are appended to a separate temporary storage location and further comprising:

deleting the temporary storage location into which the second row mapper was appending the ~~data~~ tuples from the processing of the next section.

5. (Currently Amended) The method of claim 1, wherein serially loading the ~~data~~ tuples further comprises:

loading the ~~data~~ tuples without generating SQL commands.

6. (Currently Amended) The method of claim 1, wherein the tuples output when processing data from each of the input files [[is]] are appended to a separate temporary storage location and further comprising:

when serial loading is interrupted, restarting the serial loading using the ~~data~~ tuples in the separate temporary storage locations without reprocessing the one or more input files.

7. (Original) The method of claim 1, wherein the parallel processing is performed by two or more row mappers.

8. (Currently Amended) A method for loading input data in one or more hierarchical format input files into a data store, comprising:

under control of a master row mapper,

invoking one or more slave row mappers, wherein the slave row mappers perform processing in parallel with the master row mapper and with each other, wherein the parallel processing includes parallel parsing and construction of tuples using a map specification that maps input data in the one or more input files to columns of the tuples;

processing data in a first input file to output tuples; and

~~serially loading the processed data and data~~ forwarding the tuples and tuples in one or more spillfiles to one or more database loader processes ~~into the data store~~; [[and]]

under control of each of the slave row mappers,

processing data in a separate input file to output tuples; and

storing ~~results of the processing~~ the tuples in a corresponding spillfile ; and

under control of the one or more database loader processes, serially loading the tuples into the data store.

9. (Original) The method of claim 8, further comprising:

under control of the master row mapper,

determining that there has been an error in processing the data in at least one input file; and

terminating the slave row mappers.

10. (Original) The method of claim 8, further comprising:

under control of the master row mapper,

determining that there has been an error in loading the processed data in at least one input file; and

terminating the slave row mappers.

11. (Original) The method of claim 8, further comprising:

under control of at least one of the slave row mappers,

determining that there has been an error in processing the data in at least one input file; and

terminating each of the other slave row mappers processing a separate input file whose order follows the separate input file being processed by the slave row mapper that determined that there has been an error.

12. (Currently Amended) The method of claim 8, wherein each of the one or more input files is a section, further comprising:

under control of the master row mapper and each of the slave row mappers, during processing of a current section, at the end of [[each]] a processing unit,

determining that processing has crossed into a next section; and

continuing to process data in the next section.

13. (Original) The method of claim 8, further comprising:

when restarting loading of the processed data, skipping a specified number of rows in at least one of the input files.

14. (Currently Amended) An article of manufacture comprising one of hardware logic implementing logic and a computer readable storage medium including a program for loading input data in one or more hierarchical format input files into a data store, wherein the logic or program causes operations to be performed, the operations comprising:

generating a map specification that maps input data in the one or more input files to columns of tuples;

performing parallel processing of the one or more input files to output ~~data~~ tuples, wherein the parallel processing includes parallel parsing and construction of the tuples using the map specification; and

serially loading the ~~data~~ tuples into the data store while enforcing the order of the data in the one or more input files.

15. (Original) The article of manufacture of claim 14, wherein the operations further comprise:

receiving a physical input file; and

logically dividing the physical input file into multiple sections, wherein each of the multiple sections is an input file.

16. (Original) The article of manufacture of claim 15, wherein the operations further comprise:

while performing processing of a first section from the multiple sections under control of a first row mapper,

determining that there has been an error in logically dividing the physical input file;

continuing processing of a next section from the multiple sections that is also being processed by a second row mapper; and

notifying the second row mapper to terminate processing of the next section.

17. (Currently Amended) The article of manufacture of claim 16, wherein the tuples output when processing data from each of the input files [[is]] are appended to a separate temporary storage location and wherein the operations further comprise:

deleting the temporary storage location into which the second row mapper was appending the ~~data~~ tuples from the processing of the next section.

18. (Currently Amended) The article of manufacture of claim 14, wherein the operations for serially loading the ~~data~~ tuples further comprise:

loading the ~~data~~ tuples without generating SQL commands.

19. (Currently Amended) The article of manufacture of claim 14, wherein the tuples output when processing data from each of the input files [[is]] are appended to a separate temporary storage location and wherein the operations further comprise:

when serial loading is interrupted, restarting the serial loading using the ~~data~~ tuples in the separate temporary storage locations without reprocessing the one or more input files.

20. (Original) The article of manufacture of claim 14, wherein the parallel processing is performed by two or more row mappers.

21. (Currently Amended) An article of manufacture comprising one of hardware logic implementing logic and a computer readable medium including a program for loading input data in one or more hierarchical format input files into a data store, wherein the logic or program causes operations to be performed, the operations comprising:

under control of a master row mapper,

invoking one or more slave row mappers, wherein the slave row mappers perform processing in parallel with the master row mapper and with each other, wherein the parallel processing includes parallel parsing and construction of tuples using a map specification that maps input data in the one or more input files to columns of the tuples;

processing data in a first input file to output tuples; and

~~serially loading the processed data and data~~ forwarding the tuples and tuples in one or more spillfiles to one or more database loader processes ~~into the data store~~; [[and]]

under control of each of the slave row mappers,

processing data in a separate input file to output tuples; and

storing ~~results of the processing~~ the tuples in a corresponding spillfile ; and

under control of the one or more database loader processes, serially loading the tuples into the data store.

22. (Original) The article of manufacture of claim 21, wherein the operations further comprise:

under control of the master row mapper,

determining that there has been an error in processing the data in at least one input file; and

terminating the slave row mappers.

23. (Original) The article of manufacture of claim 21, wherein the operations further comprise:

under control of the master row mapper,

determining that there has been an error in loading the processed data in at least one input file; and

terminating the slave row mappers.

24. (Original) The article of manufacture of claim 21, wherein the operations further comprise:

under control of at least one of the slave row mappers,

determining that there has been an error in processing the data in at least one input file; and

terminating each of the other slave row mappers processing a separate input file whose order follows the separate input file being processed by the slave row mapper that determined that there has been an error.

25. (Currently Amended) The article of manufacture of claim 21, wherein each of the one or more input files is a section and wherein the operations further comprise:

under control of the master row mapper and each of the slave row mappers, during processing of a current section, at the end of [[each]] a processing unit,

determining that processing has crossed into a next section; and

continuing to process data in the next section.

26. (Original) The article of manufacture of claim 21, wherein the operations further comprise:

when restarting loading of the processed data, skipping a specified number of rows in at least one of the input files.

27. (Currently Amended) A computer system having at least one program for loading input data in one or more hierarchical format input files into a data store, comprising:

generating a map specification that maps input data in the one or more input files to columns of tuples;

performing parallel processing of the one or more input files to output ~~data~~ tuples, wherein the parallel processing includes parallel parsing and construction of the tuples using the map specification; and

serially loading the ~~data~~ tuples into the data store while enforcing the order of the data in the one or more input files.

28. (Original) The computer system of claim 27, further comprising:

receiving a physical input file; and

logically dividing the physical input file into multiple sections, wherein each of the multiple sections is an input file.

29. (Original) The computer system of claim 28, further comprising:

while performing processing of a first section from the multiple sections under control of a first row mapper,

determining that there has been an error in logically dividing the physical input file;

continuing processing of a next section from the multiple sections that is also being processed by a second row mapper; and

notifying the second row mapper to terminate processing of the next section.

30. (Currently Amended) The computer system of claim 29, wherein the tuples output when processing data from each of the input files [[is]] are appended to a separate temporary storage location and further comprising:

deleting the temporary storage location into which the second row mapper was appending the ~~data~~ tuples from the processing of the next section.